



JS

APRENDE  
**JAVASCRIPT**  
GUÍA PRÁCTICA

A C A D E M I A X

# Contenido

<b>1</b>	<b>Introducción</b>	<b>4</b>
1.1	Bienvenida . . . . .	4
1.1.1	Libro vivo . . . . .	5
1.1.2	Alcance . . . . .	5
1.2	Prerequisitos . . . . .	5
1.3	¿Cómo evitar bloqueos? . . . . .	6
<b>2</b>	<b>Primeros pasos</b>	<b>7</b>
2.1	Visión general . . . . .	7
2.1.1	¿Qué es y porqué debes aprenderlo? . . . . .	7
2.1.2	¿En dónde se utiliza? . . . . .	8
2.1.3	¿Qué trabajos puedes conseguir? . . . . .	9
2.1.4	¿Cuánto puedes ganar? . . . . .	9
2.1.5	¿Cuales son las preguntas más comunes? . . . . .	10
2.2	Historia, evolución, y versiones . . . . .	11
2.3	Características y ventajas . . . . .	12
2.4	Diferencias con otros lenguajes de programación . . . . .	12
2.5	Configuración . . . . .	13
2.5.1	IDE . . . . .	13
2.5.2	Entorno . . . . .	13
2.6	Hola Mundo . . . . .	14
<b>3</b>	<b>Gramática</b>	<b>15</b>
3.1	Sintaxis . . . . .	15
3.2	Comentarios . . . . .	15
<b>4</b>	<b>Tipos y variables</b>	<b>16</b>
4.1	Variables . . . . .	16
4.2	Listas . . . . .	18

Contenido

---

4.3	Objetos . . . . .	20
4.4	Constantes . . . . .	22
<b>5</b>	<b>Operadores</b>	<b>23</b>
5.1	Operadores de aritméticos . . . . .	23
5.2	Operadores comparativos . . . . .	24
5.3	Operadores lógicos . . . . .	24
<b>6</b>	<b>Condicionales</b>	<b>25</b>
6.1	Condición única . . . . .	25
6.2	Múltiples condiciones . . . . .	26
<b>7</b>	<b>Bucles</b>	<b>28</b>
7.1	Bucle for . . . . .	28
7.2	Bucle while . . . . .	29
<b>8</b>	<b>Funciones</b>	<b>30</b>
<b>9</b>	<b>Programación orientada a objetos (POO)</b>	<b>33</b>
9.1	Paradigma . . . . .	33
9.2	Clases . . . . .	33
<b>10</b>	<b>Módulos</b>	<b>34</b>
<b>11</b>	<b>Siguientes pasos</b>	<b>36</b>
11.1	Herramientas . . . . .	36
11.2	Recursos . . . . .	36
11.3	¿Que viene después? . . . . .	37
11.4	Preguntas de entrevista . . . . .	38

# 1 Introducción

## 1.1 Bienvenida

Bienvenid@ a esta guía de Academia X en donde aprenderás JavaScript práctico.

Hola, mi nombre es Xavier Reyes Ochoa y soy el autor de esta guía. He trabajado como consultor en proyectos para Nintendo, Google, entre otros proyectos top-tier, trabajé como líder de un equipo de desarrollo por 3 años, y soy Ingeniero Ex-Amazon. En mis redes me conocen como Programador X y comparto videos semanalmente en YouTube desde diversas locaciones del mundo con el objetivo de guiar y motivar a mis estudiantes mientras trabajo haciendo lo que más me gusta: la programación.

En esta guía vas a aprender estos temas:

- Gramática
- Tipos y variables
- Operadores
- Condicionales
- Bucles
- Funciones
- Programación orientada a objetos (POO)
- Módulos

La motivación de esta guía es darte todo el conocimiento técnico que necesitas para elevar la calidad de tus proyectos y al mismo tiempo puedas perseguir metas más grandes, ya sea utilizar esta tecnología para tus pasatiempos creativos, aumentar tus oportunidades laborales, o si tienes el espíritu emprendedor, incluso crear tu propio negocio en línea. Confío en que esta guía te dará los recursos que necesitas para que tengas éxito en este campo.

Empecemos!

## 1 INTRODUCCIÓN

---

### 1.1.1 Libro vivo

Esta publicación fue planeada, editada, y revisada manualmente por Xavier Reyes Ochoa. La fundación del contenido fue generada parcialmente por inteligencia artificial usando ChatGPT (Mar 14 Version) de OpenAI. Puedes ver más detalles en <https://openai.com/>

Esto es a lo que llamo un trabajo **VIVO**, esto quiere decir que será actualizado en el tiempo a medida que existan cambios en la tecnología. La versión actual es 1.0.0 editada el 23 de marzo de 2023. Si tienes correcciones importantes, puedes escribirnos a nuestra sección de contacto en <https://www.academia-x.com>

### 1.1.2 Alcance

El objetivo de esta guía es llenar el vacío que existe sobre esta tecnología en Español siguiendo el siguiente enfoque:

- Se revizan los temas con un enfoque práctico incluyendo ejemplos y retos.
- Se evita incluir material de relleno ya que no es eficiente.
- Se evita entrar en detalle en temas simples o avanzados no-prácticos.

Si deseas profundizar en algún tema, te dejo varios recursos populares y avanzados en la lección de recursos como el estándar de esta tecnología (que puede ser difícil de leer si recién estás empezando).

## 1.2 Prerequisitos

Antes de aprender JavaScript, necesitas lo siguiente:

1. Un computador: cualquier computador moderno tiene las capacidades de correr este lenguaje. Te recomiendo un monitor de escritorio o laptop ya que dispositivos móviles o ipads no son cómodos para programar.

## 1 INTRODUCCIÓN

---

2. Sistema operativo: conocimiento básico de cómo utilizar el sistema operativo en el que se ejecutará JavaScript (por ejemplo, Windows, MacOS, Linux). Te recomiendo tener el sistema operativo actualizado.
3. Conocimiento básico de la línea de comandos: se utiliza para ejecutar programas en JavaScript.
4. Un editor de texto: lo necesitas para escribir código de JavaScript. Los editores de texto más populares son Visual Studio Code, Sublime Text, Atom y Notepad ++.
5. Un navegador web y conexión al internet: es útil para investigar más sobre esta tecnología cuando tengas dudas. Los navegadores más populares son Google Chrome, Mozilla Firefox, Safari y Microsoft Edge. Se recomienda tener el navegador actualizado.

Si ya tienes estos requisitos, estarás en una buena posición para comenzar a aprender JavaScript y profundizar en sus características y aplicaciones.

Si todavía no tienes conocimiento sobre algunos de estos temas, te recomiendo buscar tutoriales básicos en blogs a través de Google, ver videos en YouTube, o hacer preguntas a ChatGPT. Alternativamente, puedes empezar ya e investigar en línea a medida que encuentres bloqueos enteniendo los conceptos en esta guía.

### 1.3 ¿Cómo evitar bloqueos?

Para sacarle el mayor provecho a esta guía:

1. No solo leas esta guía. La práctica es esencial en este campo. Practica todos los días y no pases de lección hasta que un concepto esté 100% claro.
2. No tienes que memorizarlo todo, solo tienes que saber donde están los temas para buscarlos rápidamente cuando tengas dudas.
3. Cuando tengas preguntas usa [Google](#), [StackOverFlow](#), y [ChatGPT](#)
4. Acepta que en esta carrera, mucho de tu tiempo lo vas utilizar investigando e innovando, no solo escribiendo código.

## 2 PRIMEROS PASOS

---

5. No tienes que aprender inglés ahora pero considera aprenderlo en un futuro porque los recursos más actualizados están en inglés y también te dará mejores oportunidades laborales.
6. Si pierdes la motivación, recuerda tus objetivos. Ninguna carrera es fácil pero ya tienes los recursos para llegar muy lejos. Te deseo lo mejor en este campo!

## 2 Primeros pasos

### 2.1 Visión general

#### 2.1.1 ¿Qué es y por qué debes aprenderlo?

JavaScript es un lenguaje de programación de alto nivel y dinámico que se utiliza en el desarrollo web para agregar interactividad y dinamismo a las páginas web. JavaScript permite a los desarrolladores crear funcionalidades en las páginas web como formularios interactivos, animaciones, juegos, aplicaciones y mucho más.

Hay varias razones por las que deberías aprender JavaScript:

1. Es ampliamente utilizado: JavaScript es uno de los lenguajes de programación más populares y ampliamente utilizados en el mundo, lo que significa que hay una gran demanda de desarrolladores JavaScript.
2. Es versátil: JavaScript se utiliza tanto en el lado del cliente como en el lado del servidor, lo que significa que puedes desarrollar una amplia gama de aplicaciones y proyectos con él.
3. Es fácil de aprender: JavaScript es un lenguaje de programación fácil de aprender, especialmente para aquellos que ya tienen una comprensión básica de programación.
4. Te abre puertas: Aprender JavaScript te permitirá aprovechar nuevas oportunidades de carrera y te permitirá trabajar en una amplia gama de proyectos interesantes.

## 2 PRIMEROS PASOS

---

En resumen, JavaScript es un lenguaje de programación importante y versátil que te permitirá desarrollar una amplia gama de aplicaciones y proyectos. Aprender JavaScript puede ser una inversión valiosa en tu futuro profesional.

### 2.1.2 ¿En dónde se utiliza?

JavaScript se utiliza en una amplia variedad de contextos y plataformas, incluyendo:

1. **Desarrollo web:** JavaScript es un lenguaje de programación clave para el desarrollo web. Se utiliza para crear funcionalidades interactivas y dinámicas en las páginas web, como formularios interactivos, animaciones, galerías de imágenes, menús desplegables, juegos y mucho más.
2. **Aplicaciones web:** JavaScript se utiliza para crear aplicaciones web complejas y avanzadas, como herramientas de productividad, aplicaciones de negocios, plataformas de e-commerce y mucho más.
3. **Aplicaciones móviles:** JavaScript se utiliza en la creación de aplicaciones móviles mediante tecnologías como React Native y PhoneGap.
4. **Internet de las cosas (IoT):** JavaScript se utiliza para desarrollar aplicaciones IoT, que permiten a los dispositivos conectados a internet interactuar entre sí.
5. **Desarrollo de servidor:** JavaScript se utiliza en el lado del servidor para crear aplicaciones y servicios web. Con Node.js, puedes crear aplicaciones y servicios web complejos con JavaScript en el lado del servidor.

En resumen, JavaScript es un lenguaje de programación versátil que se utiliza en una amplia variedad de contextos y plataformas, desde el desarrollo web hasta las aplicaciones móviles y IoT.



## 2 PRIMEROS PASOS

---

### 2.1.3 ¿Qué trabajos puedes conseguir?

Aprender JavaScript puede abrirte la puerta a una amplia gama de oportunidades de carrera, incluyendo:

1. Desarrollador web front-end: Los desarrolladores web front-end utilizan JavaScript para crear y mejorar la experiencia de usuario en las páginas web.
2. Desarrollador web full-stack: Los desarrolladores web full-stack utilizan JavaScript en conjunto con otros lenguajes de programación para crear aplicaciones web complejas y avanzadas.
3. Desarrollador de aplicaciones móviles: Los desarrolladores de aplicaciones móviles utilizan JavaScript para crear aplicaciones móviles y hacerlas compatibles con diferentes plataformas.
4. Desarrollador de juegos: Los desarrolladores de juegos utilizan JavaScript para crear juegos interactivos y emocionantes para la web y dispositivos móviles.
5. Desarrollador de aplicaciones IoT: Los desarrolladores de aplicaciones IoT utilizan JavaScript para crear aplicaciones que permiten a los dispositivos conectados a internet interactuar entre sí.
6. Desarrollador de inteligencia artificial y machine learning: JavaScript se utiliza para crear aplicaciones y algoritmos de inteligencia artificial y machine learning.

En resumen, aprender JavaScript puede prepararte para una amplia gama de carreras en el campo de la tecnología, desde el desarrollo web hasta las aplicaciones móviles y la inteligencia artificial.

### 2.1.4 ¿Cuánto puedes ganar?

El salario que puedes ganar al utilizar JavaScript en tu trabajo depende de varios factores, incluyendo tu nivel de experiencia, el tipo de trabajo que desempeñes, la ubicación geográfica y la industria en la que trabajes.

## 2 PRIMEROS PASOS

---

En general, los desarrolladores web con experiencia en JavaScript pueden esperar ganar salarios competitivos en comparación con otros profesionales de la tecnología. Según Glassdoor, el salario promedio de un desarrollador web en los Estados Unidos es de aproximadamente \$75,000 al año.

Es importante tener en cuenta que estos son solo promedios y que el salario real que puedes ganar puede ser mayor o menor, dependiendo de los factores mencionados anteriormente. Además, siempre es una buena idea investigar y hacer preguntas sobre los salarios y las condiciones laborales antes de aceptar un trabajo.

### 2.1.5 ¿Cuales son las preguntas más comunes?

Algunas de las preguntas más comunes sobre JavaScript incluyen:

1. ¿Qué es JavaScript y cómo funciona?
2. ¿Qué se puede hacer con JavaScript?
3. ¿Cómo se compara JavaScript con otros lenguajes de programación?
4. ¿Cómo se instala y configura JavaScript en una computadora?
5. ¿Cómo se utiliza JavaScript para crear aplicaciones web y móviles?
6. ¿Cómo se integra JavaScript con otros lenguajes de programación y marcos de trabajo?
7. ¿Cómo se debuggear y solucionar errores en el código de JavaScript?
8. ¿Cuáles son las mejores prácticas para escribir código JavaScript eficiente y mantenible?
9. ¿Cómo se puede mejorar la seguridad de las aplicaciones de JavaScript?
10. ¿Qué recursos y comunidades pueden ayudar a aprender y mejorar en JavaScript?

Estas son solo algunas de las preguntas más comunes sobre JavaScript. Hay muchas más que pueden surgir a medida que avanzas en tu aprendizaje y desarrollo de aplicaciones con este lenguaje de programación.

## 2 PRIMEROS PASOS

---

Al finalizar este recurso, tendrás las habilidades necesarias para responder o encontrar las respuestas a estas preguntas.

### 2.2 Historia, evolución, y versiones

JavaScript es un lenguaje de programación interpretado, dinámico y orientado a objetos. Fue creado por Brendan Eich en 1995 para la empresa Netscape Communications Corporation. Originalmente fue diseñado para ser un lenguaje de scripting para la web, pero ahora se usa ampliamente en aplicaciones web, servidores y aplicaciones móviles.

La primera versión de JavaScript fue lanzada en 1996 como parte de Netscape Navigator 2.0. Desde entonces, ha habido varias versiones de JavaScript, cada una con nuevas características y mejoras. Las versiones más recientes incluyen ECMAScript 6 (ES6) y ECMAScript 7 (ES7).

Las versiones más antiguas de JavaScript se conocen como JavaScript 1.0 y JavaScript 1.1. Estas versiones fueron reemplazadas por JavaScript 1.2, que fue lanzada en 1997. Esta versión introdujo nuevas características como el soporte para objetos, la capacidad de crear objetos personalizados y la capacidad de crear funciones.

En 1999, se lanzó JavaScript 1.3, que introdujo mejoras en la velocidad de ejecución y la capacidad de crear objetos más complejos. Esta versión también introdujo el soporte para la programación orientada a objetos.

En el año 2015, se lanzó ECMAScript 6 que introdujo mejoras en la velocidad de ejecución y la capacidad de crear objetos más complejos. Esta versión también introdujo el soporte para la programación orientada a objetos.

Desde ES6, cada año se presenta una nueva versión con mejoras siendo la versión 14 para el año 2023.

## 2 PRIMEROS PASOS

---

### 2.3 Características y ventajas

JavaScript es un lenguaje de programación interpretado, orientado a objetos y multiplataforma. Está diseñado para crear aplicaciones web interactivas y dinámicas.

#### **Características:**

- Es un lenguaje de programación interpretado, lo que significa que no necesita ser compilado antes de su ejecución.
- Es orientado a objetos, lo que significa que los programas se escriben usando objetos y sus propiedades.
- Es multiplataforma, lo que significa que se puede ejecutar en diferentes sistemas operativos.
- Es compatible con la mayoría de los navegadores web, lo que significa que se puede ejecutar en la mayoría de los navegadores web.

#### **Ventajas:**

- Es fácil de aprender y usar.
- Es un lenguaje de programación versátil, lo que significa que se puede usar para crear aplicaciones web, aplicaciones de escritorio y aplicaciones móviles.
- Es un lenguaje de programación dinámico, lo que significa que los programas se pueden cambiar en tiempo de ejecución.
- Es un lenguaje de programación seguro, lo que significa que los programas se ejecutan en un entorno aislado.
- Es un lenguaje de programación escalable, lo que significa que se puede usar para crear aplicaciones de gran tamaño.

### 2.4 Diferencias con otros lenguajes de programación

JavaScript es un lenguaje de programación interpretado, lo que significa que el código se ejecuta directamente sin necesidad de compilarlo. Esto lo hace único entre

## 2 PRIMEROS PASOS

---

los lenguajes de programación, ya que la mayoría de los lenguajes de programación requieren que el código se compile antes de que se pueda ejecutar.

Otra diferencia importante entre JavaScript y otros lenguajes de programación es que JavaScript es un lenguaje orientado a objetos. Esto significa que los programadores pueden crear objetos y usarlos para almacenar y manipular datos. Esto es diferente de otros lenguajes de programación, como C, que son lenguajes estructurados y no orientados a objetos.

### 2.5 Configuración

#### 2.5.1 IDE

Los archivos de JavaScript son archivos de texto. Puedes editarlos con **editores de texto** como Notepad en Windows o Notes en MacOS pero es recomendado utilizar un **IDE** (Integrated Development Environment) que es una aplicación de edición de código más avanzado que le da colores a tu código para que sea más fácil de leer y tengas funciones de autocompletado, entre otras. Algunos IDEs populares son [Brackets](#), [Atom](#), [Sublime Text](#), [Vim](#), y [Visual Studio Code](#).

El editor recomendado para practicar el código que vamos a ver es Visual Studio Code (o VSCode) que puedes bajar desde <https://code.visualstudio.com/>

#### 2.5.2 Entorno

Para usar JavaScript en tu computador, no necesitas instalar ningún software especial, ya que JavaScript es un lenguaje interpretado y se ejecuta en el navegador web. Sin embargo, hay algunas cosas que debes tener en cuenta:

1. Navegador web: es recomendable tener un navegador web actualizado, como Google Chrome, Mozilla Firefox, Safari, etc.

## 2 PRIMEROS PASOS

---

2. Editor de texto: puedes usar un editor de texto sencillo como el Bloc de notas en Windows o TextEdit en Mac, pero también puedes usar un editor de código más avanzado como Visual Studio Code, Sublime Text, etc.
3. Conocimiento básico de HTML y CSS: ya que JavaScript se utiliza para agregar dinamismo y interactividad a las páginas web, es importante tener una comprensión básica de HTML y CSS.

Una vez que tengas estos elementos, estarás list@ para escribir y ejecutar código JavaScript en tu navegador. Puedes crear un archivo HTML y agregar tu código JavaScript dentro de una etiqueta **<script>** o puedes escribir código JavaScript directamente en la consola del navegador.

### 2.6 Hola Mundo

“Hola Mundo” es un ejemplo clásico que se utiliza para mostrar el funcionamiento básico de un lenguaje de programación.

Ejemplo:

En este ejemplo, se imprime el texto “Hola Mundo” en la consola.



También podrías modificar este código con tu nombre. Si es “Juan”, debería imprimir en la consola “Hola, Juan” .

Reto:

Modifica el ejemplo anterior para imprimir “Hola Universo” en la consola.

## 3 GRAMÁTICA

---

### 3 Gramática

#### 3.1 Sintaxis

La sintaxis de un lenguaje de programación es la estructura de un lenguaje de programación, que incluye reglas para la construcción de programas. Estas reglas se refieren a la forma en que los elementos de un programa se relacionan entre sí, como los operadores, variables, palabras clave, etc. La sintaxis también se refiere a la forma en que los programas se escriben, como la indentación y la estructura de los bloques de código.

Las palabras clave en JavaScript son palabras reservadas que tienen un significado especial para el intérprete de JavaScript. Estas palabras clave incluyen if, else, switch, for, while, do, break, continue, function, return, etc.

La indentación es una buena práctica en JavaScript para mejorar la legibilidad del código. Esto significa que los bloques de código se deben indentar para indicar su relación con otros bloques de código.

El conjunto de caracteres en JavaScript es Unicode, lo que significa que se pueden usar caracteres de diferentes idiomas.

JavaScript es sensible a mayúsculas y minúsculas, lo que significa que las palabras clave, variables y nombres de funciones deben escribirse de forma exacta para que el intérprete de JavaScript las reconozca.

#### 3.2 Comentarios

Los comentarios en JavaScript son líneas de texto que no son interpretadas como parte del código. Se usan para proporcionar información adicional sobre el código, como explicaciones, notas, o instrucciones para el desarrollador. Los comentarios también se pueden usar para deshabilitar código temporalmente, sin tener que eliminarlo completamente.

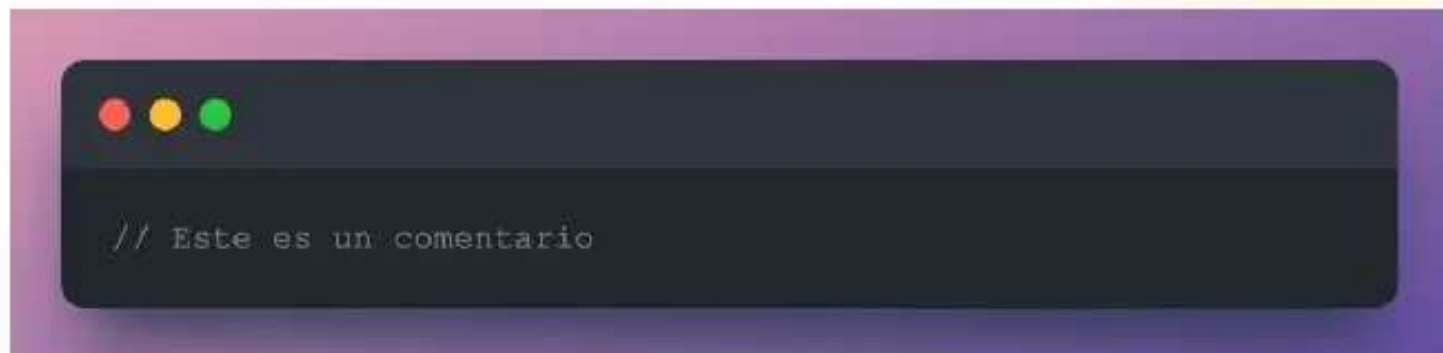


## 4 TIPOS Y VARIABLES

---

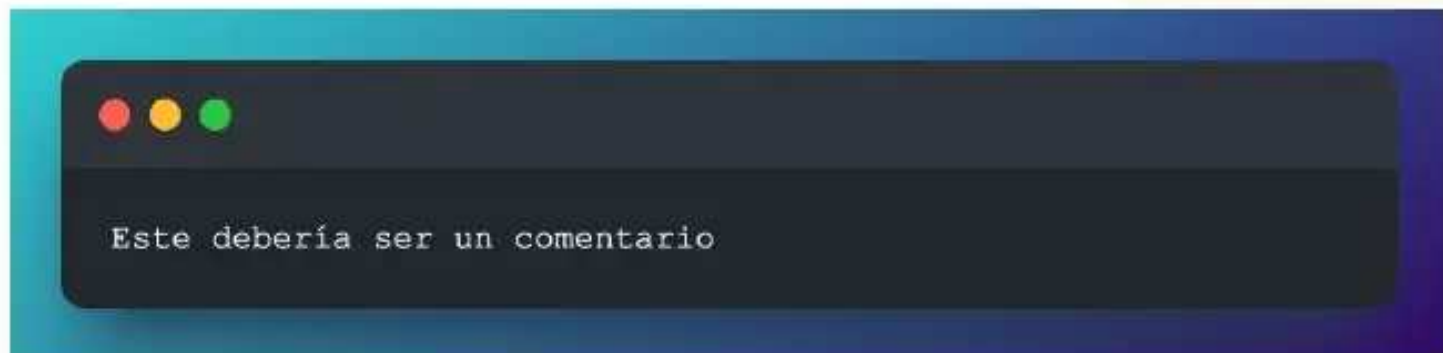
Ejemplo:

Este código es un comentario.



Reto:

Comenta esta línea para que no cause errores y se lea como comentario:



## 4 Tipos y variables

### 4.1 Variables

La manipulación de datos es una tarea fundamental de un lenguaje de programación. Para trabajar con datos necesitamos guardarlos en variables.

Una variable es un contenedor para almacenar datos que retiene su nombre y puede cambiar su valor a lo largo del tiempo. En los siguientes ejemplos vamos a ver varios tipos de datos que puedes guardar en variables.



## 4 TIPOS Y VARIABLES

---

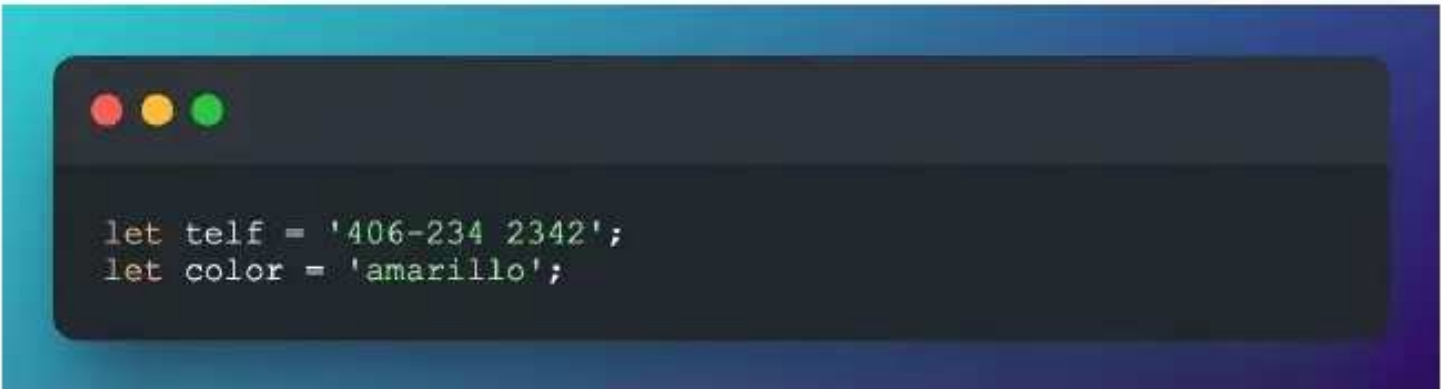
Ejemplo:

Este código declara una variable llamada “libro” y le asigna el valor de una cadena de texto que contiene el título de un libro. Luego, imprime el valor de la variable “libro” en la consola.



```
let libro = 'El Programador Pragmático';  
console.log(libro);
```

Texto es un tipo de dato útil para guardar información como números telefónicos y colores, entre otros. Este código asigna dos variables, una llamada telf que contiene un número de teléfono como una cadena de caracteres, y otra llamada color que contiene el color amarillo como una cadena de caracteres.

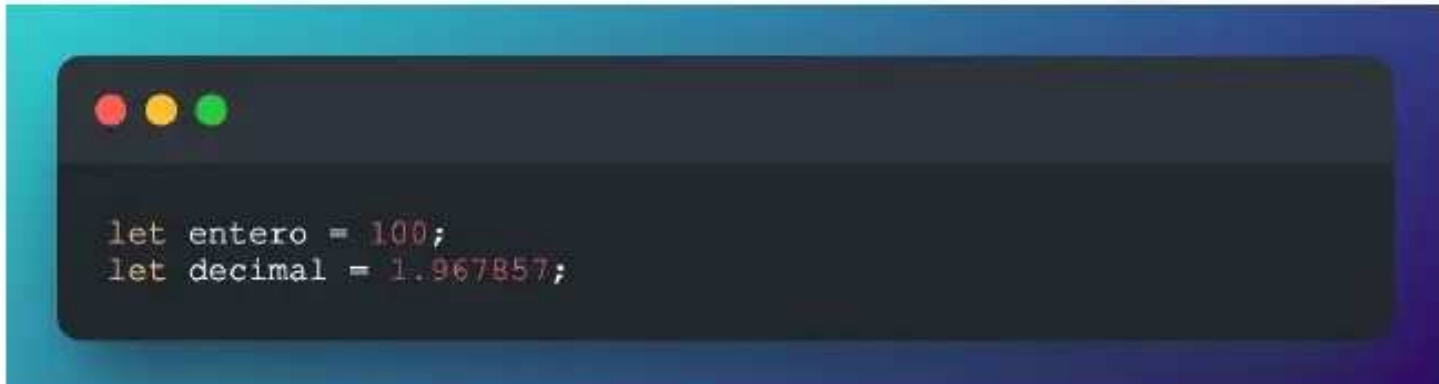


```
let telf = '406-234 2342';  
let color = 'amarillo';
```

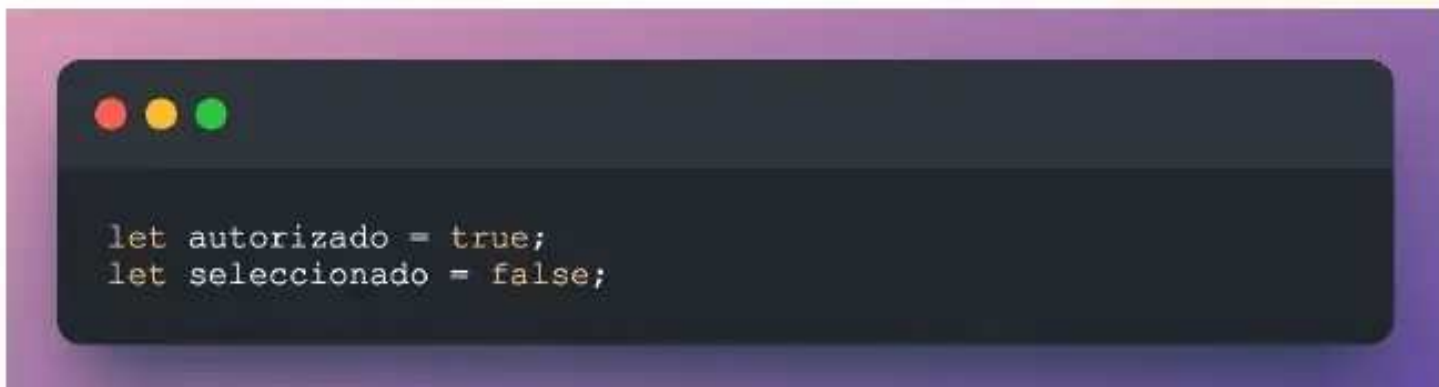
También podemos guardar datos como números enteros y decimales. Estos datos se usan para realizar operaciones matemáticas y representar valores de peso, dinero, entre otros. Este código asigna dos variables, una con un valor entero (100) y otra con un valor decimal (1.967857).

## 4 TIPOS Y VARIABLES

---



El tipo de dato booleano representa los valores de verdadero y falso. Este tipo de datos es útil, por ejemplo, para indicar si un usuario está autorizado a acceder a una app o no, entre varios usos. Este código crea una variable llamada “autorizado” que es verdadera y otra variable llamada “seleccionado” que es falsa.



Es importante saber que, en el mundo del código binario, el número 1 representa verdadero y 0 representa falso.

Reto:

Crea una variable “a” con 33 como texto e imprímela a la consola.

### 4.2 Listas

Las listas en JavaScript son una estructura de datos que nos permite almacenar una colección ordenada de elementos. Estos elementos pueden ser de cualquier tipo, desde números hasta sublistas. También los vas a encontrar con otros nombres

## 4 TIPOS Y VARIABLES

---

como arreglos, matrices, arrays, etc.

Ejemplo:

Este código está imprimiendo el primer elemento de una matriz de números a la consola. En este caso, el primer elemento es 23.



```
let numeros = [23, 45, 16, 37, 3, 99, 22];  
console.log(numeros[0]); // 23
```

También existen listas de texto. Este código crea una variable llamada “animales” que contiene una lista de elementos, en este caso, tres animales: perro, gato y tigre.



```
let animales = ['perro', 'gato', 'tigre'];
```

Y esta es una lista de datos mixtos. Este código crea una variable llamada “datosMixtos” que contiene una lista de elementos de diferentes tipos, como texto, números, booleanos y otra lista.

## 4 TIPOS Y VARIABLES

---

A code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. It contains the following JavaScript code:

```
let datosMixtos = ['texto', 69, true, ['lista dentro de otra lista']];
```

```
let datosMixtos = ['texto', 69, true, ['lista dentro de otra lista']];
```

Reto:

Crea una lista en JavaScript que contenga los nombres de los meses del año. Accede al 3er índice e imprímelo en la consola.

### 4.3 Objetos

Los objetos en JavaScript son una forma de almacenar y organizar datos mapeándolos de uno a uno. Estos datos se almacenan en forma de pares clave-valor, donde la clave suele ser una cadena de texto y el valor puede ser cualquier tipo de dato. También los vas a encontrar con otros nombres como mapas, diccionarios, etc.


Ejemplo:

Este código crea un objeto llamado jugadores que contiene dos pares clave-valor. El primer par clave-valor es 10: 'Messi' y el segundo es 7: 'Cristiano Ronaldo'.

Luego, se imprime el valor asociado con la clave 10, que es Messi.

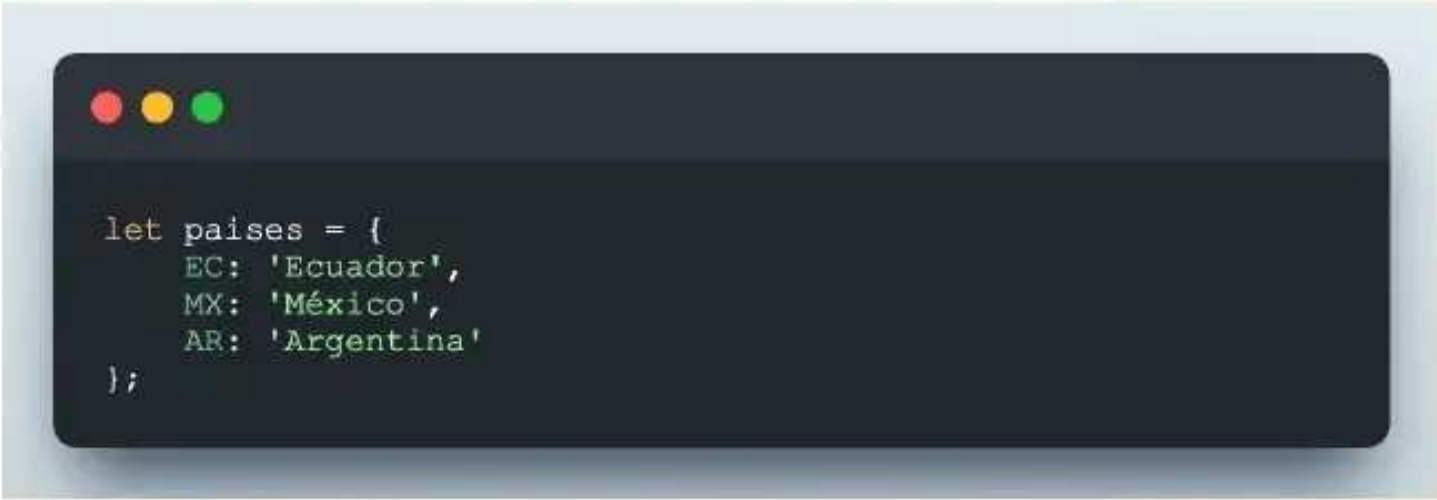
## 4 TIPOS Y VARIABLES

---



```
let jugadores = {  
  10: 'Messi',  
  7: 'Cristiano Ronaldo'  
};  
console.log(jugadores[10]); // 'Messi'
```

También podemos mapear de texto a texto. Este código crea un objeto llamado “países” que contiene claves y valores. Las claves son códigos de países (EC, MX, AR) y los valores son los nombres de los países (Ecuador, México, Argentina).




```
let paises = {  
  EC: 'Ecuador',  
  MX: 'México',  
  AR: 'Argentina'  
};
```

E incluso podemos mapear de texto a listas de texto, entre muchas otras opciones. Este código establece un objeto llamado “emails” que contiene dos pares clave-valor. Cada clave es un nombre de persona y el valor es un arreglo de direcciones de correo electrónico asociadas con esa persona.

## 4 TIPOS Y VARIABLES

---



```
let emails = {  
  'Juan': ['juan@gmail.com'],  
  'Ricardo': ['ricardo@gmail.com', 'rick@aol.com']  
};
```

Reto:


Crea un objeto en JavaScript que represente una computadora, con sus respectivas características (marca, modelo, procesador, memoria RAM, etc).

### 4.4 Constantes

Las constantes son variables que no pueden ser reasignadas. Esto significa que una vez que se asigna un valor a una constante, este no puede ser cambiado.

Ejemplo:

Esta línea de código establece una constante llamada “pi” con un valor de 3.14159265359. Esta constante se puede usar para almacenar un valor numérico que no cambiará a lo largo del programa.



```
const pi = 3.14159265359;
```

Reto:

## 5 OPERADORES

---

Crea una constante llamada 'SALUDO' y asígnale el valor 'Hola Planeta' . Luego, imprime el valor de la constante en la consola.

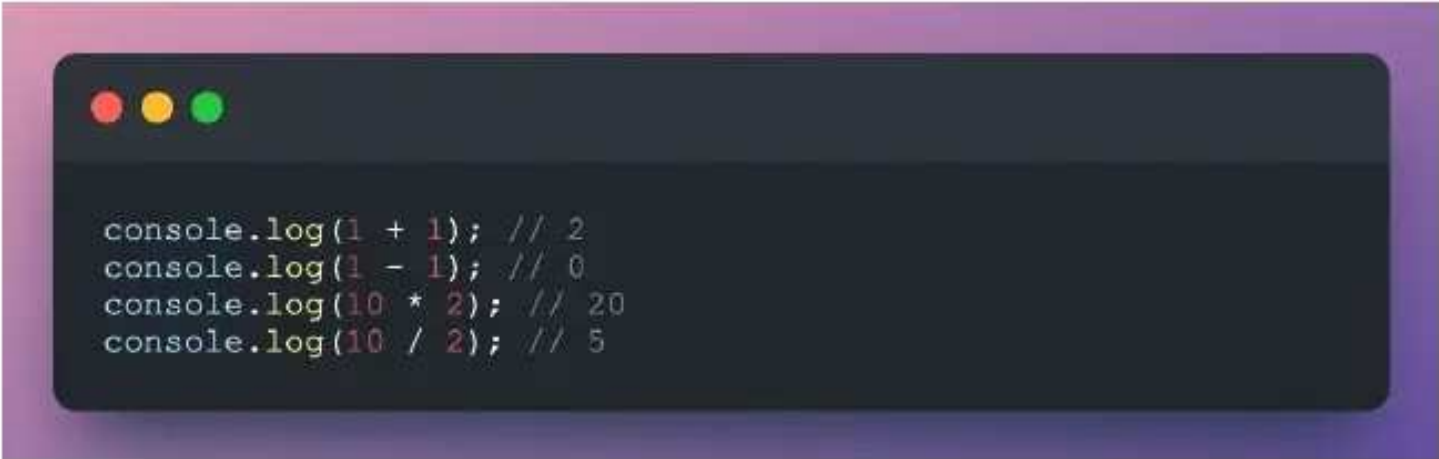
## 5 Operadores

### 5.1 Operadores de aritméticos

Los operadores aritméticos en JavaScript son usados para realizar operaciones matemáticas básicas. Estos operadores incluyen sumar, restar, multiplicar, dividir, entre otros.

Ejemplo:

Este código realiza operaciones matemáticas básicas, imprimiendo los resultados en la consola. Estas operaciones incluyen suma, resta, multiplicación y división.



```
console.log(1 + 1); // 2
console.log(1 - 1); // 0
console.log(10 * 2); // 20
console.log(10 / 2); // 5
```

Reto:

Usa los operadores aritméticos para calcular el área de un círculo con radio 5.



## 5 OPERADORES

---

### 5.2 Operadores comparativos

Los operadores comparativos en JavaScript son usados para comparar dos valores y determinar si son iguales o diferentes. Estos operadores son útiles para determinar si dos valores son iguales, si dos valores no son iguales, si un valor es mayor o menor que otro, entre otros.

Ejemplo:

Este código muestra cómo se usan los operadores para comparar valores.

A screenshot of a code editor window with a dark background and light-colored text. The code consists of seven lines, each using a different comparison operator to compare the number 4 with various values. The operators used are ==, ===, !==, <, and >=, each followed by a comment indicating the boolean result of the comparison.

```
console.log(4 == 4); // true
console.log(4 == '4'); // true
console.log(4 === '4'); // false
console.log(4 !== 5); // true
console.log(4 < 5); // true
console.log(4 >= 5); // false
```

Reto:

Escribe un programa que compare dos números 'a' y 'b' y determina si 'a' con el valor de 4 es mayor, menor o igual a 'b' con un valor de 2.

### 5.3 Operadores lógicos

Los operadores lógicos en JavaScript se usan para realizar comparaciones entre valores y devolver un valor booleano (verdadero o falso). Estos operadores pueden ser útiles, por ejemplo, si deseamos saber si un 'animal' es un gato y es un mamífero (al mismo tiempo).

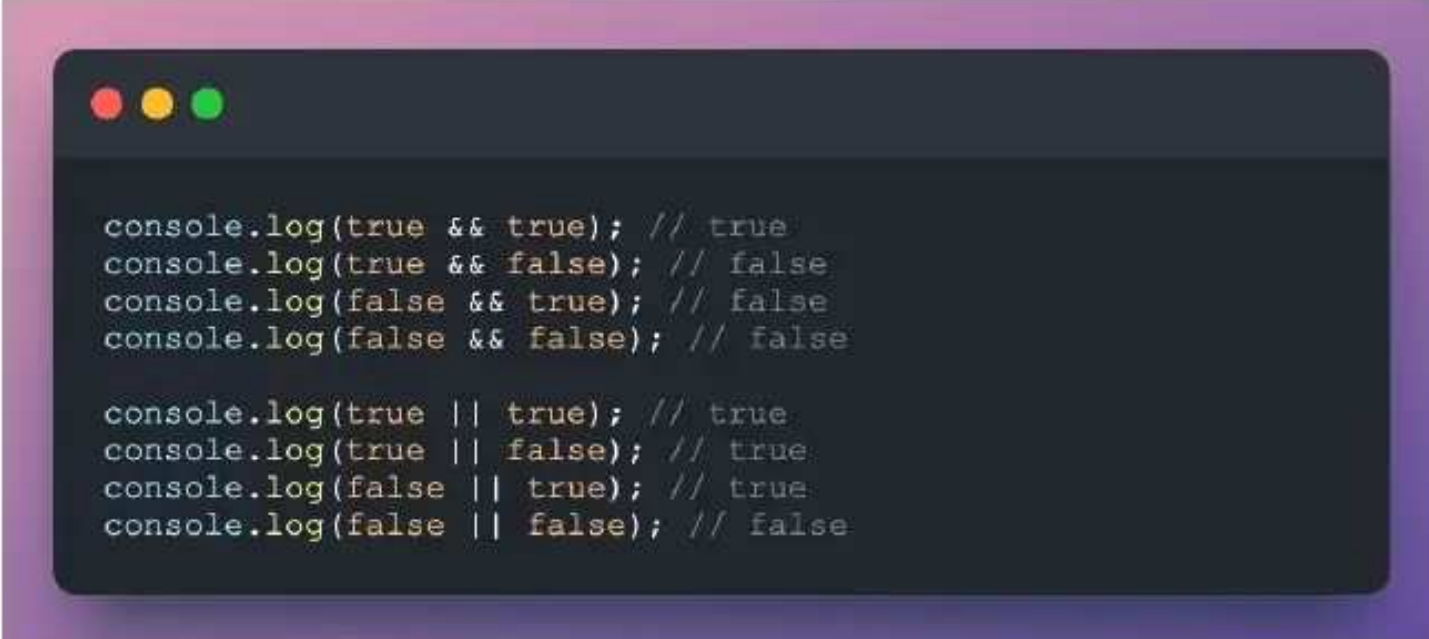
Ejemplo:



## 6 CONDICIONALES

---

Este código muestra el uso de los operadores lógicos ‘y’ y ‘o’ para evaluar expresiones booleanas. El operador ‘y’ devuelve verdadero si ambas expresiones son verdaderas, de lo contrario devuelve falso. El operador ‘o’ devuelve verdadero si al menos una de las expresiones es verdadera, de lo contrario devuelve falso.

A screenshot of a code editor window with a dark background and light-colored text. The code demonstrates the use of logical AND (&&) and OR (||) operators. The first block shows four console.log statements for the AND operator, and the second block shows four for the OR operator. Each statement includes a comment indicating the expected boolean result.

```
console.log(true && true); // true
console.log(true && false); // false
console.log(false && true); // false
console.log(false && false); // false

console.log(true || true); // true
console.log(true || false); // true
console.log(false || true); // true
console.log(false || false); // false
```

Reto:

Escribe una línea de código que devuelva verdadero si ‘x’ es mayor que 0 y ‘y’ es menor que 0.

## 6 Condicionales

### 6.1 Condición única


Los condicionales son estructura de control de flujo en JavaScript, es decir, controlan el flujo del código. Permiten ejecutar una sección de código si una condición es verdadera. También permiten ejecutar otra sección de código, si la condición es falsa.

## 6 CONDICIONALES

---

Ejemplo:

Este código comprueba si una variable booleana (autorizado) es verdadera. Si es así, imprime un mensaje en la consola indicando que el usuario puede ingresar. Si no es así, imprime un mensaje indicando que el usuario no puede ingresar.

A screenshot of a code editor window with a dark background and light-colored text. The code is in JavaScript and uses an if-else statement to check the value of a variable named 'autorizado'. If it is true, it logs 'Puede ingresar' to the console. If it is false, it logs 'No puede ingresar'. The code is as follows:

```
let autorizado = true;

if (autorizado) {
  console.log('Puede ingresar'); // ✓
} else {
  console.log('No puede ingresar');
}
```

Reto:

Escribe código condicional que revise si un número 'a' es par o impar. La variable 'a' tiene el valor de 17 y debe imprimir a la consola 'Es par' si es par o 'Es impar' si es impar.

### 6.2 Múltiples condiciones

Adicionalmente, los condicionales permiten ejecutar varias secciones de código de acuerdo a varias condiciones.

Ejemplo:

En este caso podemos ver que el código que se ejecuta depende de varios valores. Este código comprueba si una variable llamada entero es igual a 99 o 100. Si es igual a 99, imprime "Es 99" en la consola. Si es igual a 100, imprime "Es 100" en la consola. Si no es ni 99 ni 100, imprime "No 99 ni 100" en la consola.

## 6 CONDICIONALES

---

```
let entero = 100;

if (entero === 99) {
  console.log('Es 99');
} else if (entero === 100) {
  console.log('Es 100'); // ✓
} else {
  console.log('No 99 ni 100');
}
```

Este condicional es útil cuando hay una gran cantidad de posibles resultados para una expresión. Este código compara una variable (color) con diferentes valores y ejecuta una acción dependiendo del resultado de la comparación. En este caso, si la variable color es igual a 'amarillo', se imprimirá 'Advertencia' en la consola.

```
let color = 'amarillo';

switch (color) {
  case 'verde':
    console.log('Éxito');
    break;
  case 'amarillo':
    console.log('Advertencia'); // ✓
    break;
  default:
    console.log('Error');
    break;
}
```

## 7 BUCLES

---

Reto:

Crea un programa que evalúe una variable 'numero' y dependiendo del resultado, imprima un mensaje diferente. Si el número es mayor a 10, imprime "El número es mayor a 10" . Si el número es menor a 10, imprime "El número es menor a 10" . Si el número es igual a 10, imprime "El número es igual a 10" .


## 7 Bucles

### 7.1 Bucle for

Los bucles son otra estructura de control de flujo que se utilizan para iterar sobre una secuencia de elementos. También se los llama loops o ciclos.

Ejemplo:

Este código itera sobre una lista de animales e imprime cada elemento de la lista en la consola.



```
let animales = ['perro', 'gato', 'tigre'];  
  
for (let animal of animales) {  
  console.log(animal);  
}  
  
// 'perro'  
// 'gato'  
// 'tigre'
```

Reto:

## 7 BUCLES

---


Escribe un bucle que imprima los números del 1 al 10 en orden ascendente.

### 7.2 Bucle while

while es un tipo de bucle en JavaScript que se usa para ejecutar una serie de instrucciones mientras se cumpla una condición. Esta condición se evalúa antes de cada iteración del bucle.

Ejemplo:

Este código establece un bucle while que imprime los números enteros desde 100 hasta 911 en la consola. El bucle while se ejecutará mientras la variable entero sea menor o igual a la variable emergencia. Cada vez que el bucle se ejecuta, la variable entero se incrementa en 1.



```
let entero = 100;
let emergencia = 911;

while (entero <= emergencia) {
  console.log(entero);
  // entero = entero + 1;
  entero++;
}

// 100 - 911
```

Ten cuidado de no incluir una condición para parar el bucle. Esto podría seguir corriendo tu código indefinidamente hasta congelar tu computador.

Reto:

Crea un bucle while que imprima los números del 1 al 10 en la consola.

## 8 FUNCIONES

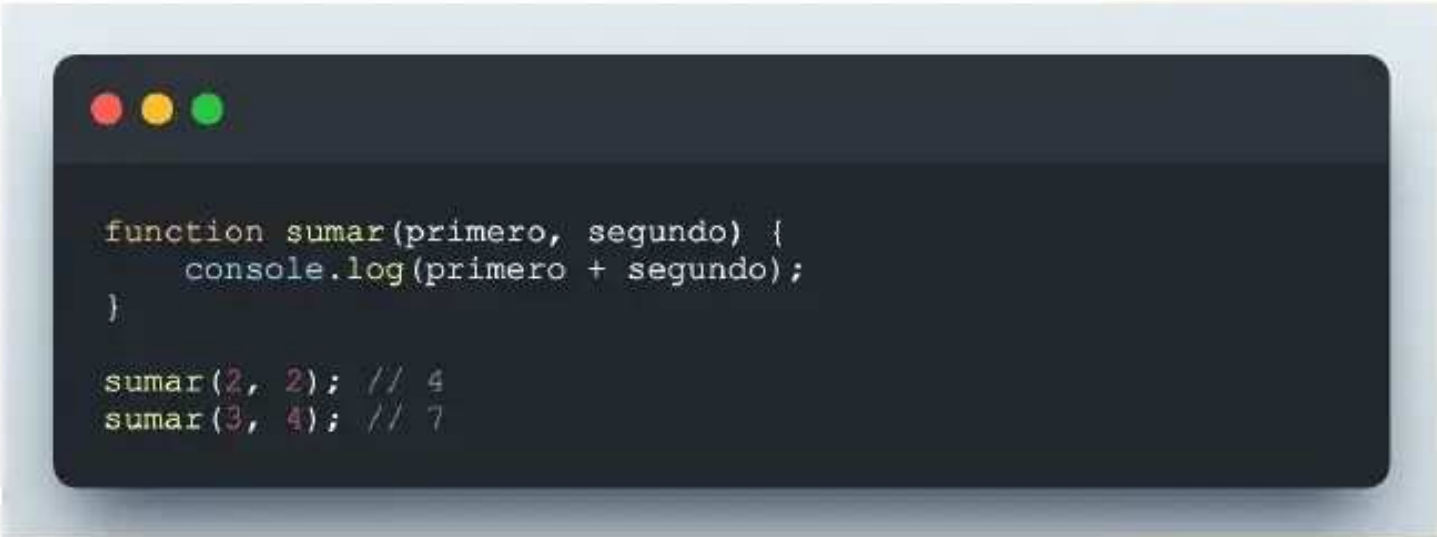
---

### 8 Funciones

En JavaScript, una función es un bloque de código diseñado para realizar una tarea específica y se puede reutilizar en varias partes el código. Las funciones se pueden definir para realizar cualquier tarea, desde realizar cálculos hasta mostrar mensajes en la pantalla.

Ejemplo:

Esta función toma dos argumentos (primero y segundo) de forma dinámica y los suma. Luego, imprime el resultado en la consola. Esta función se llama dos veces con diferentes argumentos para mostrar los resultados de la suma.

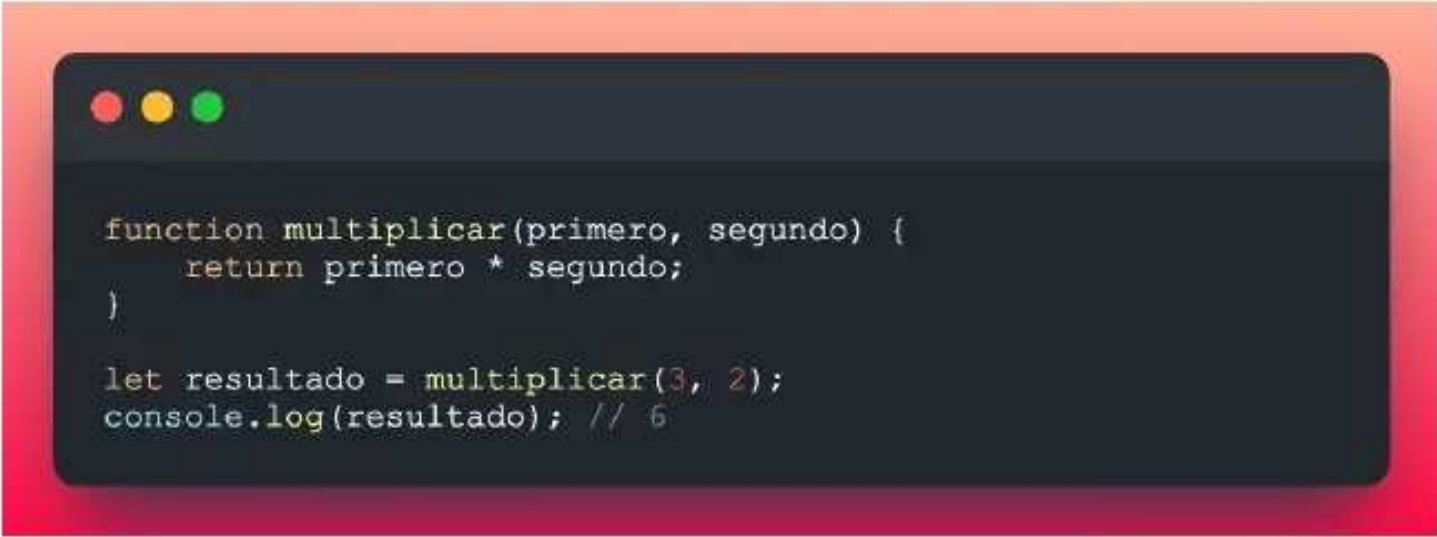
A code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The code is written in a light-colored font with syntax highlighting. It defines a function 'sumar' that takes two arguments, 'primero' and 'segundo', and logs their sum to the console. Below the function definition, there are two function calls: 'sumar(2, 2); // 4' and 'sumar(3, 4); // 7'.

```
function sumar(primero, segundo) {  
    console.log(primero + segundo);  
}  
  
sumar(2, 2); // 4  
sumar(3, 4); // 7
```

También puedes crear funciones que retornen un valor. Esto significa que una vez que se ejecuta una función, el valor devuelto se puede usar en otra parte del código. Esta función toma dos argumentos (primero y segundo) y los multiplica para devolver el resultado. En este caso, el resultado es 6.

## 8 FUNCIONES


---



```
function multiplicar(primer, segundo) {  
    return primero * segundo;  
}  
  
let resultado = multiplicar(3, 2);  
console.log(resultado); // 6
```

Esta función imprime el primer elemento de una lista. En este caso, la lista es una lista de animales, y la función imprime el primer elemento de la lista, que es 'perro'

Funciones como esta son útiles para evitar la repetición de código y para ahorrar tiempo.



```
function imprimirPrimerElemento(lista) {  
    console.log(lista[0]);  
}  
  
let animales = ['perro', 'gato', 'tigre'];  
imprimirPrimerElemento(animales); // 'perro'
```

Finalmente, puedes ver otro ejemplo de una función bastante compleja. Esta función implementa el algoritmo de ordenamiento QuickSort para ordenar una lista de elementos. El algoritmo comienza tomando un elemento de la lista como pivote y luego coloca los elementos menores que el pivote a su izquierda y los elementos mayores a su derecha. Luego, se aplica el mismo algoritmo a las sublistas izquierda y derecha hasta que la lista esté ordenada. No tienes que entender todo lo que hace



## 8 FUNCIONES

---

internamente pero te dará una idea de la complejidad que pueden tener programas como apps con miles de funciones.



```
function quicksort(lista) {  
  if (lista.length <= 1) {  
    return lista;  
  }  
  let pivote = lista[0];  
  let izquierda = [];  
  let derecha = [];  
  for (let i = 1; i < lista.length; i++) {  
    lista[i] < pivote ? izquierda.push(lista[i]) :  
derecha.push(lista[i]);  
  }  
  return quicksort(izquierda).concat(pivote,  
quicksort(derecha));  
};  
  
let numeros = [23, 45, 16, 37, 3, 99, 22];  
let listaOrdenada = quicksort(numeros);  
console.log(listaOrdenada); // [ 3, 16, 22, 23, 37, 45, 99 ]
```

Reto:

Escribe una función que tome dos números como argumentos y devuelva el mayor de los dos.



## 9 PROGRAMACIÓN ORIENTADA A OBJETOS (POO)

---

# 9 Programación orientada a objetos (POO)

### 9.1 Paradigma

La Programación Orientada a Objetos (POO) es un paradigma de programación que se basa en la creación de objetos que contienen datos y funcionalidades. Estos objetos se relacionan entre sí para formar una estructura de datos compleja. Los lenguajes de programación orientados a objetos permiten que los programadores puedan crear objetos y usarlos para construir aplicaciones.

### 9.2 Clases


Las clases en JavaScript son una forma de definir objetos creando una plantilla. Se pueden usar para crear objetos con propiedades y métodos similares. Las propiedades son valores que pertenecen al objeto y los métodos son funciones que pertenecen al objeto.

Ejemplo:

En este ejemplo creamos una clase llamada 'Lenguaje' que inicializamos con las propiedades 'nombre' y 'año'. Adicionalmente creamos el método 'descripción' que imprime un texto usando las propiedades previas. Con esa clase podemos crear una instancia para el lenguaje 'HTML' y otra para 'CSS'. Ahora podemos crear, en teoría, un número infinito de lenguajes sin reescribir los objetos manualmente.

## 10 MÓDULOS

---



```
class Lenguaje {
  constructor(nombre, año) {
    this.nombre = nombre;
    this.año = año;
  }
  descripcion() {
    console.log(`${this.nombre} fue creado en
    ${this.año}`);
  }
}

let html = new Lenguaje('HTML', 1993);
let css = new Lenguaje('CSS', 1996);
html.descripcion(); // 'HTML fue creado en 1993'
css.descripcion(); // 'CSS fue creado en 1996'
```

Reto:

Crea una clase llamada Auto que tenga propiedades como marca, modelo y año. Luego, crea un par de instancias de esta clase.

## 10 Módulos

Los módulos son una forma de incluir código externo de otro archivo. Esto permite a los desarrolladores reutilizar código y ahorrar tiempo al escribir código.

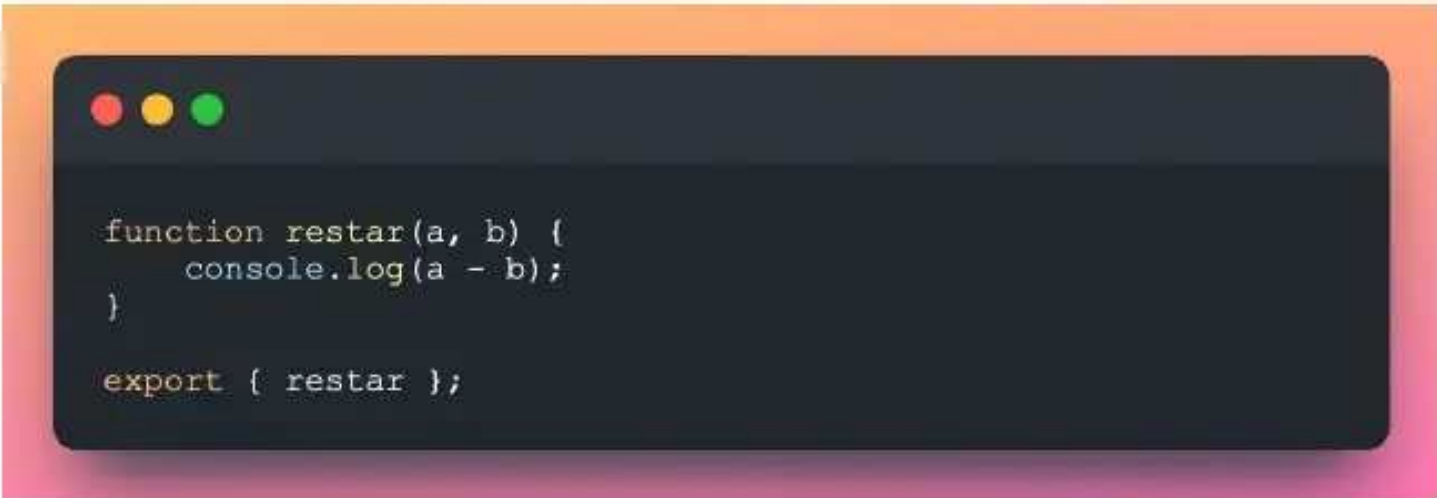
Ejemplo:

Este archivo define una función que toma dos argumentos (a y b) y luego imprime el resultado de la resta de a y b en la consola. El código exporta la función para que pueda ser usada en otros archivos. Exportar es una forma de compartir código

## 10 MÓDULOS

---

entre archivos. Esto significa que un archivo puede exportar variables, funciones y objetos para que otros archivos los puedan usar.



```
function restar(a, b) {  
  console.log(a - b);  
}  
  
export { restar };
```

Y en otro archivo podemos importar y utilizar esta función. Este código importa una función desde el archivo externo y luego la usa para restar 10 menos 2, resultando en 8.



```
import { restar } from './restar.js';  
restar(10, 2); // 8
```

Reto:

Crea un archivo llamado que contenga una función para calcular el área de un rectángulo. Luego, importa la función en otro archivo y usala para calcular el área de un rectángulo con lados de 3 y 4.

## 11 SIGUIENTES PASOS

---

# 11 Siguientes pasos

## 11.1 Herramientas

1. **Node.js:** Es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome. Proporciona una plataforma para construir aplicaciones de servidor y aplicaciones de línea de comandos con JavaScript.
2. **React.js:** Es una biblioteca de JavaScript de código abierto para crear interfaces de usuario. Está diseñado para ayudar a los desarrolladores a construir aplicaciones web y móviles escalables.
3. **Angular.js:** Es un marco de JavaScript de código abierto para desarrollar aplicaciones web. Está diseñado para ayudar a los desarrolladores a construir aplicaciones web modernas y escalables.
4. **jQuery:** Es una biblioteca de JavaScript de código abierto para simplificar la interacción con el HTML, el CSS y el DOM. Está diseñado para simplificar la programación de JavaScript.
5. **TypeScript:** Es un superconjunto de JavaScript que proporciona una sintaxis más estructurada y una mejor administración de errores. Está diseñado para ayudar a los desarrolladores a escribir código JavaScript más robusto y escalable.

## 11.2 Recursos

1. MDN Web Docs: Es una de las fuentes de referencia más completas para JavaScript. Ofrece documentación detallada sobre la sintaxis, los conceptos y las API de JavaScript. También ofrece tutoriales y ejemplos para ayudar a los desarrolladores a aprender y usar JavaScript. <https://developer.mozilla.org/es/docs/Web/JavaScript>

## 11 SIGUIENTES PASOS

---

2. W3Schools: Es una de las fuentes de referencia más populares para JavaScript. Ofrece tutoriales, ejemplos y referencias para ayudar a los desarrolladores a aprender y usar JavaScript. <https://www.w3schools.com/js/>
3. Stack Overflow: Es una de las comunidades de desarrolladores más grandes en línea. Ofrece una gran cantidad de preguntas y respuestas sobre JavaScript, así como una sección de discusión para discutir problemas y soluciones relacionadas con JavaScript. <https://stackoverflow.com/questions/tagged/javascript>
4. TutorialesPoint: Ofrece tutoriales, ejemplos y referencias para ayudar a los desarrolladores a aprender y usar JavaScript. <https://www.tutorialspoint.com/javascript/index.htm>
5. Codecademy: Ofrece una variedad de cursos interactivos para ayudar a los desarrolladores a aprender y usar JavaScript. <https://www.codecademy.com/learn/introduction-to-javascript>

### 11.3 ¿Que viene después?

Estos son los siguientes pasos recomendados:

1. Practicar JavaScript: Una vez que hayas aprendido los conceptos básicos de JavaScript, es importante practicar para mejorar tus habilidades. Puedes hacer esto escribiendo código en un editor de texto y ejecutándolo en un navegador web.
2. Aprender un marco de JavaScript: Un marco de JavaScript es un conjunto de herramientas y librerías que te ayudan a desarrollar aplicaciones web más rápido. Algunos de los marcos más populares son React, Angular y Vue.
3. Aprender una biblioteca de JavaScript: Una biblioteca de JavaScript es un conjunto de funciones y herramientas que te ayudan a realizar tareas comunes de desarrollo web. Algunas de las bibliotecas más populares son jQuery, Lodash y Underscore.

## 11 SIGUIENTES PASOS

---

4. Aprender una herramienta de desarrollo de JavaScript: Una herramienta de desarrollo de JavaScript es una herramienta que te ayuda a escribir, depurar y optimizar tu código. Algunas de las herramientas más populares son Webpack, Babel y Gulp.
5. Aprender una base de datos relacional: Una base de datos relacional es una base de datos que almacena datos en tablas relacionadas entre sí. Algunas de las bases de datos relacionales más populares son MySQL, PostgreSQL y SQLite.
6. Aprender una base de datos no relacional: Una base de datos no relacional es una base de datos que almacena datos en formato de documentos. Algunas de las bases de datos no relacionales más populares son MongoDB, CouchDB y Redis.

### 11.4 Preguntas de entrevista

1. ¿Qué es JavaScript?
  - JavaScript es un lenguaje de programación interpretado, orientado a objetos, que se utiliza principalmente para crear contenido interactivo en páginas web.
2. ¿Qué es una variable en JavaScript?
  - Una variable es un contenedor para almacenar datos. En JavaScript, una variable se declara usando la palabra clave “var” .
3. ¿Cómo se declara una variable en JavaScript?
  - Una variable se declara usando la palabra clave “var” seguida del nombre de la variable y, opcionalmente, el valor que se le asignará.
4. ¿Qué es una función en JavaScript?

## 11 SIGUIENTES PASOS

---

- Una función es un bloque de código que se puede ejecutar cuando se le llama. Las funciones se utilizan para realizar tareas específicas y se pueden reutilizar en diferentes partes de un programa.
5. ¿Cómo se define una función en JavaScript?
    - Una función se define usando la palabra clave “function” seguida del nombre de la función y los parámetros que se pasarán a la función.
  6. ¿Qué es un objeto en JavaScript?
    - Un objeto es una colección de propiedades y métodos relacionados. Los objetos se utilizan para representar entidades reales o abstractas en un programa.
  7. ¿Cómo se crea un objeto en JavaScript?
    - Un objeto se crea usando la palabra clave “new” seguida del nombre del constructor del objeto. El constructor es una función especial que se utiliza para inicializar un objeto.
  8. ¿Qué es una cadena en JavaScript?
    - Una cadena es una secuencia de caracteres. En JavaScript, las cadenas se escriben entre comillas simples o dobles.
  9. ¿Qué es una matriz en JavaScript?
    - Una matriz es una colección ordenada de valores. En JavaScript, las matrices se escriben entre corchetes.
  10. ¿Qué es un bucle en JavaScript?
    - Un bucle es una estructura de control que se utiliza para ejecutar un bloque de código repetidamente. En JavaScript, los bucles se escriben usando la palabra clave “for” .
  11. ¿Qué es una condición en JavaScript?

## 11 SIGUIENTES PASOS

---

- Una condición es una expresión que se evalúa como verdadera o falsa. En JavaScript, las condiciones se escriben usando la palabra clave `“if”` .
12. ¿Qué es una expresión regular en JavaScript?
- Una expresión regular es un patrón de caracteres que se utiliza para buscar y reemplazar cadenas de texto. En JavaScript, las expresiones regulares se escriben entre barras.
13. ¿Qué es una clase en JavaScript?
- Una clase es una plantilla para crear objetos. En JavaScript, las clases se definen usando la palabra clave `“class”` .
14. ¿Qué es una excepción en JavaScript?
- Una excepción es un error que se produce durante la ejecución de un programa. En JavaScript, las excepciones se manejan usando la palabra clave `“try”` .
15. ¿Qué es una etiqueta en JavaScript?
- Una etiqueta es una palabra clave que se utiliza para marcar una sección de código. En JavaScript, las etiquetas se escriben entre corchetes.
16. ¿Qué es una propiedad en JavaScript?
- Una propiedad es una característica de un objeto. En JavaScript, las propiedades se definen usando la palabra clave `“this”` .
17. ¿Qué es un método en JavaScript?
- Un método es una función asociada a un objeto. En JavaScript, los métodos se definen usando la palabra clave `“this”` .
18. ¿Qué es una sentencia en JavaScript?



## 11 SIGUIENTES PASOS

---

- Una sentencia es una instrucción que se ejecuta cuando se alcanza un punto específico en un programa. En JavaScript, las sentencias se escriben usando la palabra clave `“return”`.
19. ¿Qué es una declaración en JavaScript?
- Una declaración es una instrucción que se ejecuta cuando se alcanza un punto específico en un programa. En JavaScript, las declaraciones se escriben usando la palabra clave `“var”`.
20. ¿Qué es una expresión en JavaScript?
- Una expresión es una combinación de valores, variables y operadores que se evalúa como un valor. En JavaScript, las expresiones se escriben usando la palabra clave `“return”`.
21. ¿Qué es un operador en JavaScript?
- Un operador es un símbolo que se utiliza para realizar una operación matemática o lógica. En JavaScript, los operadores se escriben usando la palabra clave `“operator”`.
22. ¿Qué es una sentencia de control en JavaScript?
- Una sentencia de control es una instrucción que se utiliza para controlar el flujo de un programa. En JavaScript, las sentencias de control se escriben usando la palabra clave `“switch”`.
23. ¿Qué es una sentencia de iteración en JavaScript?
- Una sentencia de iteración es una instrucción que se utiliza para ejecutar un bloque de código repetidamente. En JavaScript, las sentencias de iteración se escriben usando la palabra clave `“for”`.
24. ¿Qué es una sentencia de salto en JavaScript?

## 11 SIGUIENTES PASOS

---

- Una sentencia de salto es una instrucción que se utiliza para saltar a una parte específica de un programa. En JavaScript, las sentencias de salto se escriben usando la palabra clave `“break”`.

25. ¿Qué es una sentencia de etiqueta en JavaScript?

- Una sentencia de etiqueta es una instrucción que se utiliza para marcar una sección de código. En JavaScript, las sentencias de etiqueta se escriben usando la palabra clave `“label”`.

26. ¿Qué es una sentencia de continuación en JavaScript?

- Una sentencia de continuación es una instrucción que se utiliza para saltar a la siguiente iteración de un bucle. En JavaScript, las sentencias de continuación se escriben usando la palabra clave `“continue”`.

27. ¿Qué es una sentencia de asignación en JavaScript?

- Una sentencia de asignación es una instrucción que se utiliza para asignar un valor a una variable. En JavaScript, las sentencias de asignación se escriben usando la palabra clave `“=”`.

28. ¿Qué es una sentencia de declaración en JavaScript?

- Una sentencia de declaración es una instrucción que se utiliza para declarar una variable. En JavaScript, las sentencias de declaración se escriben usando la palabra clave `“var”`.

29. ¿Qué es una sentencia de función en JavaScript?

- Una sentencia de función es una instrucción que se utiliza para definir una función. En JavaScript, las sentencias de función se escriben usando la palabra clave `“function”`.

30. ¿Qué es una sentencia de bloque en JavaScript?

## 11 SIGUIENTES PASOS

---

- Una sentencia de bloque es una instrucción que se utiliza para agrupar un conjunto de sentencias. En JavaScript, las sentencias de bloque se escriben usando la palabra clave `“block”`.
31. ¿Qué es una sentencia de etiqueta en JavaScript?
- Una sentencia de etiqueta es una instrucción que se utiliza para marcar una sección de código. En JavaScript, las sentencias de etiqueta se escriben usando la palabra clave `“label”`.
32. ¿Qué es una sentencia de excepción en JavaScript?
- Una sentencia de excepción es una instrucción que se utiliza para manejar errores. En JavaScript, las sentencias de excepción se escriben usando la palabra clave `“try”`.
33. ¿Qué es una sentencia de clase en JavaScript?
- Una sentencia de clase es una instrucción que se utiliza para definir una clase. En JavaScript, las sentencias de clase se escriben usando la palabra clave `“class”`.
34. ¿Qué es una sentencia de objeto en JavaScript?
- Una sentencia de objeto es una instrucción que se utiliza para crear un objeto. En JavaScript, las sentencias de objeto se escriben usando la palabra clave `“new”`.
35. ¿Qué es una sentencia de importación en JavaScript?
- Una sentencia de importación es una instrucción que se utiliza para importar código de otro archivo. En JavaScript, las sentencias de importación se escriben usando la palabra clave `“import”`.
36. ¿Qué es una sentencia de exportación en JavaScript?

## 11 SIGUIENTES PASOS

---

- Una sentencia de exportación es una instrucción que se utiliza para exportar código a otro archivo. En JavaScript, las sentencias de exportación se escriben usando la palabra clave `“export”`.
37. ¿Qué es una sentencia de declaración de módulo en JavaScript?
- Una sentencia de declaración de módulo es una instrucción que se utiliza para definir un módulo. En JavaScript, las sentencias de declaración de módulo se escriben usando la palabra clave `“module”`.
38. ¿Qué es una sentencia de declaración de paquete en JavaScript?
- Una sentencia de declaración de paquete es una instrucción que se utiliza para definir un paquete. En JavaScript, las sentencias de declaración de paquete se escriben usando la palabra clave `“package”`.
39. ¿Qué es una sentencia de declaración de interfaz en JavaScript?
- Una sentencia de declaración de interfaz es una instrucción que se utiliza para definir una interfaz. En JavaScript, las sentencias de declaración de interfaz se escriben usando la palabra clave `“interface”`.
40. ¿Qué es una sentencia de declaración de enumeración en JavaScript?
- Una sentencia de declaración de enumeración es una instrucción que se utiliza para definir una enumeración. En JavaScript, las sentencias de declaración de enumeración se escriben usando la palabra clave `“enum”`.
41. ¿Qué es una sentencia de declaración de tipo en JavaScript?
- Una sentencia de declaración de tipo es una instrucción que se utiliza para definir un tipo. En JavaScript, las sentencias de declaración de tipo se escriben usando la palabra clave `“type”`.
42. ¿Qué es una sentencia de declaración de constante en JavaScript?

## 11 SIGUIENTES PASOS

---

- Una sentencia de declaración de constante es una instrucción que se utiliza para definir una constante. En JavaScript, las sentencias de declaración de constante se escriben usando la palabra clave `“const”`.
43. ¿Qué es una sentencia de declaración de variable en JavaScript?
- Una sentencia de declaración de variable es una instrucción que se utiliza para definir una variable. En JavaScript, las sentencias de declaración de variable se escriben usando la palabra clave `“var”`.
44. ¿Qué es una sentencia de declaración de función en JavaScript?
- Una sentencia de declaración de función es una instrucción que se utiliza para definir una función. En JavaScript, las sentencias de declaración de función se escriben usando la palabra clave `“function”`.
45. ¿Qué es una sentencia de declaración de clase en JavaScript?
- Una sentencia de declaración de clase es una instrucción que se utiliza para definir una clase. En JavaScript, las sentencias de declaración de clase se escriben usando la palabra clave `“class”`.
46. ¿Qué es una sentencia de declaración de objeto en JavaScript?
- Una sentencia de declaración de objeto es una instrucción que se utiliza para definir un objeto. En JavaScript, las sentencias de declaración de objeto se escriben usando la palabra clave `“object”`.
47. ¿Qué es una sentencia de declaración de método en JavaScript?
- Una sentencia de declaración de método es una instrucción que se utiliza para definir un método. En JavaScript, las sentencias de declaración de método se escriben usando la palabra clave `“method”`.
48. ¿Qué es una sentencia de declaración de propiedad en JavaScript?

## 11 SIGUIENTES PASOS

---

- Una sentencia de declaración de propiedad es una instrucción que se utiliza para definir una propiedad. En JavaScript, las sentencias de declaración de propiedad se escriben usando la palabra clave “property” .
49. ¿Qué es una sentencia de declaración de evento en JavaScript?
- Una sentencia de declaración de evento es una instrucción que se utiliza para definir un evento. En JavaScript, las sentencias de declaración de evento se escriben usando la palabra clave “event” .
50. ¿Qué es una sentencia de declaración de excepción en JavaScript?
- Una sentencia de declaración de excepción es una instrucción que se utiliza para definir una excepción. En JavaScript, las sentencias de declaración de excepción se escriben usando la palabra clave “exception” .
51. ¿Qué es una sentencia de declaración de interfaz en JavaScript?
- Una sentencia de declaración de interfaz es una instrucción que se utiliza para definir una interfaz. En JavaScript, las sentencias de declaración de interfaz se escriben usando la palabra clave “interface” .